

Yun Shield Quick Start Guide

VERSION: 1.0

Version	Description	Date
1.0	Release	2014-Jul-08
1.1	Change Password to dragino	2014-Aug-02

Index:

1	Introduction	3
1.1	About this quick start guide	3
1.2	What is Yun Shield	3
1.3	Specifications	3
1.4	Features	4
1.5	System Structure	5
2	Set up and use Yun Shield	7
2.1	Connect to Leonardo and power.....	7
2.2	Connect to Yun Shield.....	7
2.3	Set Up Yun Shield to access internet	8
2.4	Detect Yun Shield.....	8
2.5	Upload Sketch.....	9
2.6	Bridge Library	9
3	Examples	10
3.1	Example 1: Say hello to Linux.....	10
3.2	Example 2: Upload data to IoT Server	12

1 Introduction

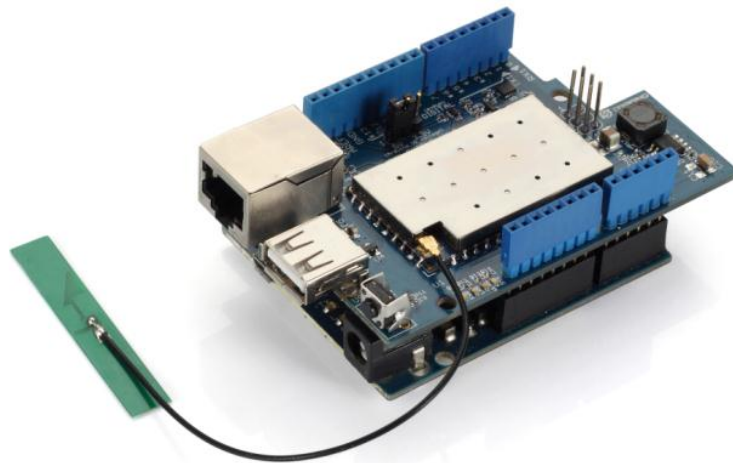
1.1 About this quick start guide

This is a quick start guide to introduce what is Yun Shield and how to use Yun Shield with Arduino Leonardo and run simple Bridge examples. For more detail about Yun Shield system other features or use with other Arduino boards. Please refer the Yun Shield User Manual.

1.2 What is Yun Shield

Yun Shield is one of the most powerful shields for Arduino Board. Yun Shield is designed to solve the Internet connectivity and storage issue for Arduino Board.

Yun Shield runs Open Source OpenWrt system (Same system as runs in Arduino Yun) and it is fully compatible with Arduino IDE v1.5.4 or later. Yun Shield is the ideally choice for Arduino Projects which require various internet connections and more storage.



Basically, Yun Shield + Leonardo equally to the official Arduino Yun, but Yun Shield is more flexible because it can work with other Arduino board such as Uno, Duemilanove, Mega etc. And Yun Shield uses external wifi antenna which provides stability and possibility for various environments.

1.3 Specifications

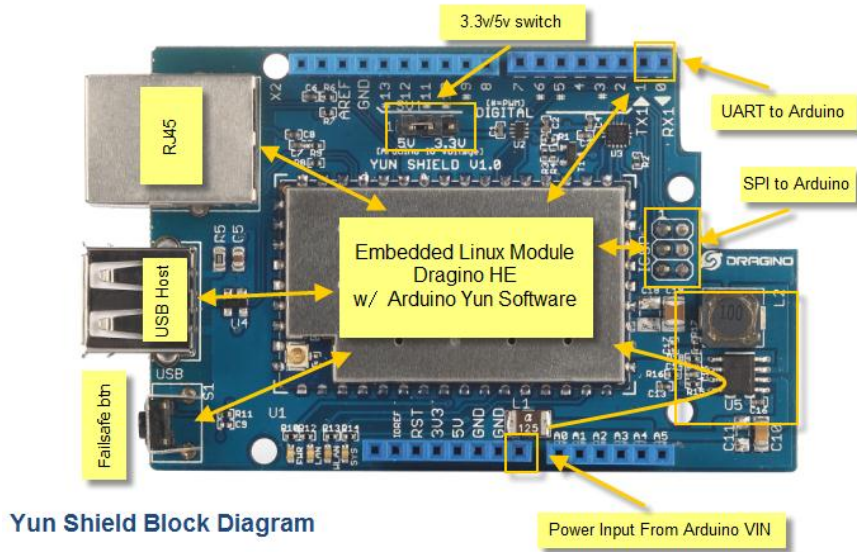
- Processor: 400MHz, 24K MIPS
- Flash: 16MBytes
- RAM: 64MBytes
- Power Input: 4.75v ~ 23v via Arduino VIN pin
- 1 x 10M/100M RJ45 connector
- 150M WiFi 802.11 b/g/n
- External Antenna via I-Pex connector
- 1 x USB 2.0 host connector, used for USB storage or 3G connection

- 1 x Reset button
- Compatible with 3.3v or 5v I/O Arduino.

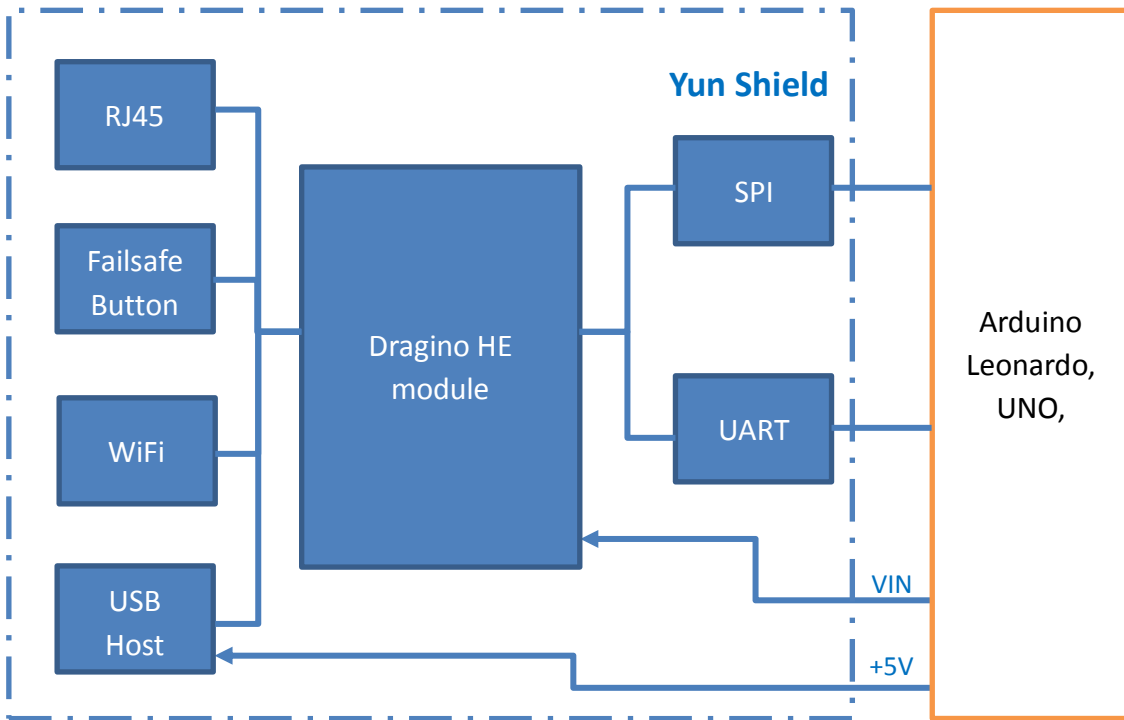
1.4 Features

- ✓ Open source Linux (OpenWrt) inside
- ✓ Low power consumption
- ✓ Compatible with Arduino IDE 1.5.4 or later, user can program, debug or upload sketch to Arduino board via Arduino IDE.
- ✓ Managed by Web GUI, SSH via LAN or WiFi
- ✓ Software upgradable via network
- ✓ Built-in web server
- ✓ Support internet connection via LAN port, WiFi or 3G dongle.
- ✓ Support USB flash to provide storage for Arduino projects.
- ✓ Failsafe design provides robustly system.
- ✓ Compatible with Arduino Leonardo, Uno , Duemilanove, Diecimila, Mega

1.5 System Structure



Yun Shield Block Diagram



POWER:

The Dragino HE is the core module of Yun Shield. The HE module requires around 200ma current when in full load, so it is powered by the Arduino **VIN** pins to avoid overheated in the Arduino onboard 5v LDO. So when Yun shield is in used, the Arduino board should be powered by DC port instead of USB port. The DC input can be **7v ~ 15v**.

The USB Host of Yun Shield gets power from the Arduino **+5v** pin, since the +5v from Arduino comes from the +5V LDO, to avoid overheated on the Arduino Board, when the USB host is in used, it is recommended to use **+7v DC**.

Interface:

The RJ45, WiFi, USB Host and Failsafe are connected to the Dragino HE module directly. And the Dragino HE module use SPI and UART to communicate with Arduino Board. Yun Shield is compatible with 3.3v and 5v Arduino board. The **on board jumper SV1** is used to set the SPI and UART to 3.3v or 5v level.

The SPI interface is used to upload the sketches comes from the Arduino IDE. SPI interface only connects to Dragino HE during uploading so the Arduino SPI can still be used to connect to other SPI slave devices.

The UART interface is used for the Bridge class in Arduino, there are lots of examples explain how to use the bridge class in the Arduino IDE. It is the core of Yun solution. We must make sure the serial Interface of Arduino is not used by other hardware.

2 Set up and use Yun Shield

2.1 Connect to Leonardo and power

Simply put the Yun Shield on the top of Arduino Leonardo and Power the Leonardo via the **DC Jack**.

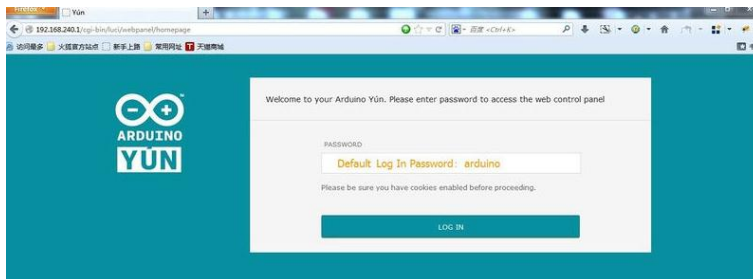
2.2 Connect to Yun Shield



At the first boot of Yun Shield, it will auto generate an unsecure WiFi network call *Dragino2-xxxxxx*

User can use their laptop to connect to this WiFi network. The laptop will get an IP 192.168.240.xxx and the Yun Shield has the default IP **192.168.240.1**

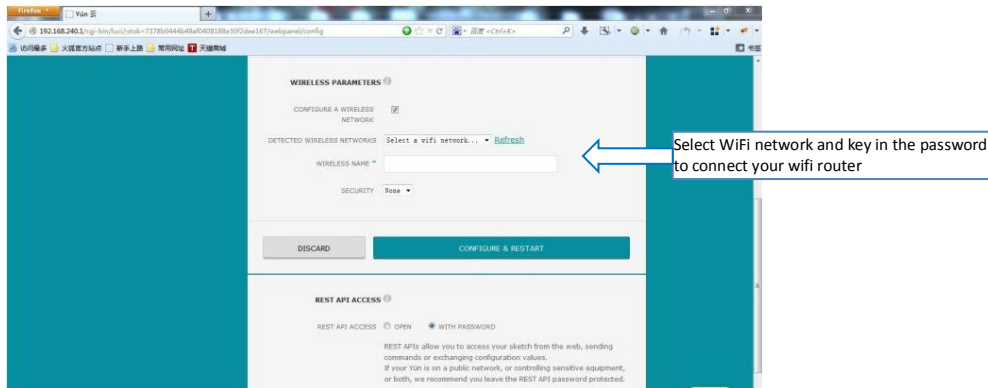
Once user joins the network, they can use web browser (recommend **Firefox** and **Chrome**) and enter 192.168.240.1 to enter the Yun Shield setting page.



Default Password for Yun Shield is **dragino**.

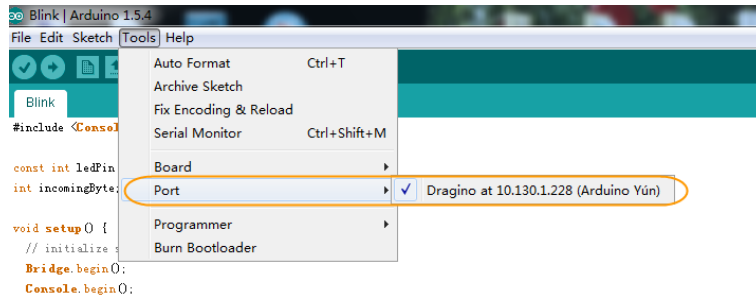
2.3 Set Up Yun Shield to access internet

After log in, the GUI will show the WIFI / ETH interface status. Click the Configure button and now user can configure Yun Shield to access internet via your WiFi router.



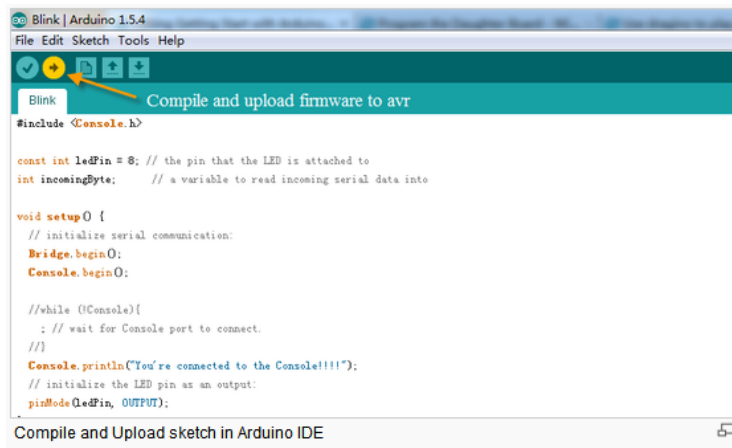
2.4 Detect Yun Shield

Assume your laptop and Yun Shield are in the same network. The Yun Shield will broadcast data in this network and Arduino IDE will receive this data and show the Yun Shield in **Tools** → **Port**.



2.5 Upload Sketch

- 1) In the Arduino IDE, choose the **Arduino Yun** board type for Leonardo.
- 2) In Arduino IDE → port, choose the correct port. (should be Arduino Yun port with an ip address)
- 3) In the Yun Shield GUI → Sensor page, choose the Board Type: Leonardo
- 4) Compile the sketch and upload it to the Arduino Board. During upload, The Yun Shield will ask you to key in the password, by default, the password is **dragino**.



2.6 Bridge Library

The Bridge Library simplifies the communication between the Arduino Board and Yun Shield.

Bridge commands from the AVR (Arduino Board) are interpreted by Python on the Yun Shield. Its role is to execute programs on the GNU/Linux side when asked by Arduino, provide a shared storage space for sharing data like sensor readings between the Arduino and the Internet, and receiving commands from the Internet and passing them directly to the Arduino.

There are detail explain and lots of example to show how to use Bridge in the Arduino Official Website. Reference link is: <http://arduino.cc/en/Reference/YunBridgeLibrary> and there are two examples in next chapter.

3 Examples

3.1 Example 1: Say hello to Linux

Introduction:

This example is a hello test between the Arduino and Yun Shield. The example can be found on the [Arduino IDE--> File --> Examples --> Bridge --> ConsoleRead](#). Tutorial of this example can be found on <http://arduino.cc/en/Tutorial/ConsoleRead>. Below listing the code and add some detail to understand it with the Yun Shield:

Code:

```
#include <Console.h> //use Console class for Arduino IDE debug over WiFi, similar to Serial class,
String name;

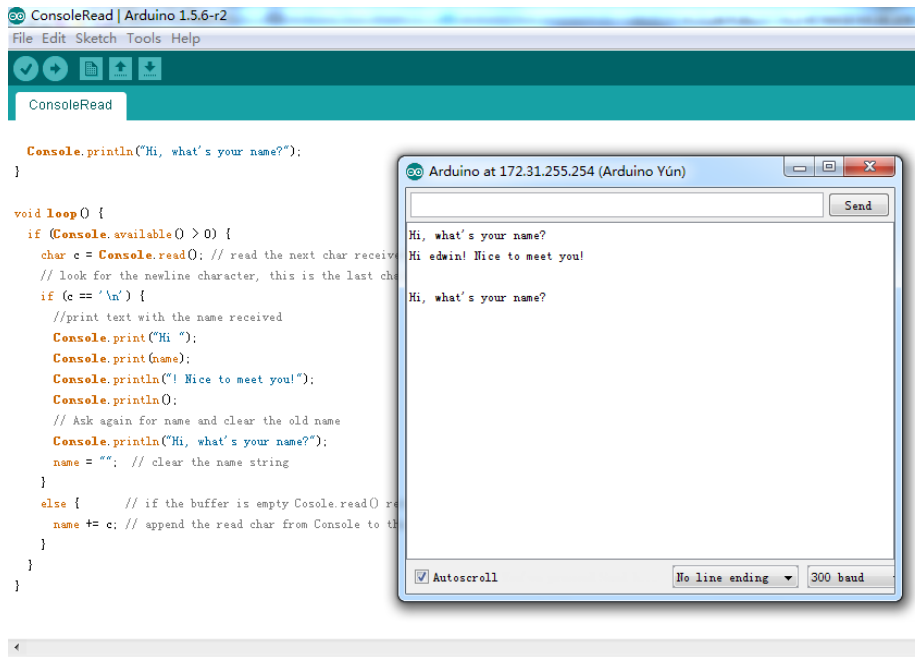
void setup() {
  // Initialize Console and wait for port to open:
  Bridge.begin();
  Console.begin();

  // Wait for Console port to connect
  while (!Console);

  Console.println("Hi, what's your name?"); //Data flow: Arduino --> Yun Shield --> Arduino IDE
}

void loop() {
  if (Console.available() > 0) {
    char c = Console.read(); //read the next char received, data flow: IDE --> Yun Shield--> Arduino
    // look for the newline character, this is the last character in the string
    if (c == '\n') {
      //print text with the name received
      Console.print("Hi ");
      Console.print(name);
      Console.println("! Nice to meet you!");
      Console.println();
      // Ask again for name and clear the old name
      Console.println("Hi, what's your name?");
      name = ""; // clear the name string
    }
    else { // if the buffer is empty Cosole.read() returns -1
      name += c; // append the read char from Console to the name string
    }
  }
}
```

Screen Shot:



The screenshot displays the Arduino IDE interface. The sketch editor shows the following code:

```
Console.println("Hi, what's your name?");
}

void loop() {
  if (Console.available() > 0) {
    char c = Console.read(); // read the next char received
    // look for the newline character, this is the last char
    if (c == '\n') {
      //print text with the name received
      Console.print("Hi ");
      Console.print(name);
      Console.println("! Nice to meet you!");
      Console.println();
      // Ask again for name and clear the old name
      Console.println("Hi, what's your name?");
      name = ""; // clear the name string
    }
    else { // if the buffer is empty Console.read() returns '\0'
      name += c; // append the read char from Console to the name string
    }
  }
}
```

The serial monitor window shows the following output:

```
Hi, what's your name?
Hi edwin! Nice to meet you!
Hi, what's your name?
```

The serial monitor settings are: Autoscroll (checked), No line ending, and 300 baud.

3.2 Example 2: Upload data to IoT Server

Introduction:

This example shows how to log data to the public IoT server “Xively”. The example is a modified version (change Serial to Console to fit for different Arduino Board and debug over WiFi) from [Arduino IDE--> File --> Examples --> Bridge --> XivelyClient](#). Tutorial of this example can refer <http://arduino.cc/en/Tutorial/YunXivelyClient>.

Before upload the sketch, make sure:

- ✓ The Yun Shield already has internet access
- ✓ Input your FEED ID and API KEY according to the Tutorial. Note, The FEED ID should be within double quotation marks “”.
- ✓ Change Serial Class to Console class to fit for different AVRs.

Below listing the code and add some detail to understand it with the Yun Shield:

Code:

```
// include all Libraries needed:
#include <Process.h>           //Process lib use to call Linux Commands in Yun Shield
#include <Console.h>          //Console lib, used to show debug info in Arduino IDE
#include "passwords.h"        // contains my passwords, see below

/*
 NOTE: passwords.h is not included with this repo because it contains my passwords.
 You need to create it for your own version of this application. To do so, make
 a new tab in Arduino, call it passwords.h, and include the following variables and constants:

#define APIKEY           "foo"                // replace your pachube api key here
#define FEEDID           "0000"              // replace your feed ID
#define USERAGENT       "my-project"        // user agent is the project name
*/

// set up net client info:
const unsigned long postingInterval = 60000; //delay between updates to xively.com
unsigned long lastRequest = 0;               // when you last made a request
String dataString = "";

void setup() {
  // start console:
  Bridge.begin();
  Console.begin();

  while (!Console); // wait for Network Serial to open
  Console.println("Xively client");

  // Do a first update immediately
  updateData();
  sendData();
  lastRequest = millis();
}

void loop() {
  // get a timestamp so you can calculate reading and sending intervals:
  long now = millis();
```

```
// if the sending interval has passed since your
// last connection, then connect again and send data:
if (now - lastRequest >= postingInterval) {
    updateData();
    sendData();
    lastRequest = now;
}
}

void updateData() {
    // convert the readings to a String to send it:
    dataString = "Temperature,";
    dataString += random(10) + 20;
    // add pressure:
    dataString += "\nPressure,";
    dataString += random(5) + 100;
}

// this method makes a HTTP connection to the server:
void sendData() {
    // form the string for the API header parameter:
    String apiString = "X-ApiKey: ";
    apiString += APIKEY;

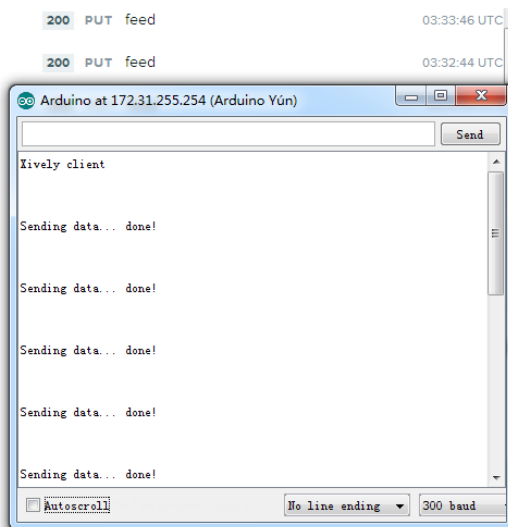
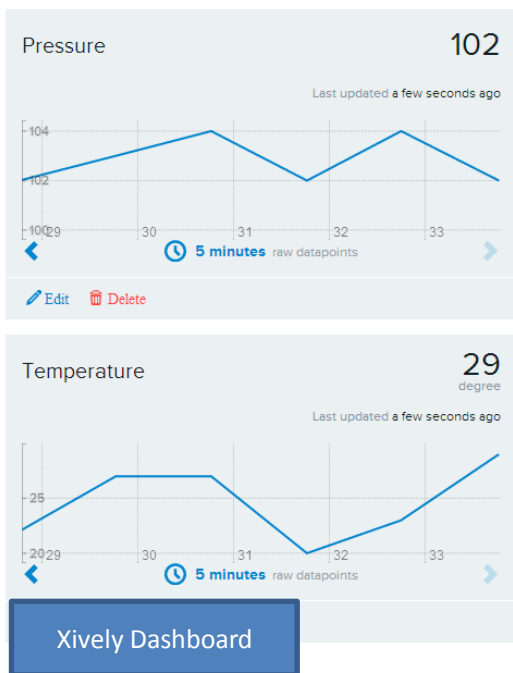
    // form the string for the URL parameter:
    String url = "https://api.xively.com/v2/feeds/";
    url += FEEDID;
    url += ".csv";

    // Send the HTTP PUT request, form the linux command and use Process Class to send this command to Yun
    Shield

    // Is better to declare the Process here, so when the
    // sendData function finishes the resources are immediately
    // released. Declaring it global works too, BTW.
    Process xively;
    Console.print("\n\nSending data... ");
    xively.begin("curl");
    xively.addParameter("-k");
    xively.addParameter("--request");
    xively.addParameter("PUT");
    xively.addParameter("--data");
    xively.addParameter(dataString);
    xively.addParameter("--header");
    xively.addParameter(apiString);
    xively.addParameter(url);
    xively.run();
    Console.println("done!");

    // If there's incoming data from the net connection,
    // send it out the Console:
    while (xively.available() > 0) {
        char c = xively.read();
        Console.write(c);
    }
}
}
```

Screen Shot:



Triggers

Triggers provide 'push' capabilities by sending HTTP POST requests to a URL of your choice when a condition has been satisfied.